“应用程序缓存”：和一般的浏览器缓存不同，它不会随着用户清理浏览器缓存而被清除。

目的：使Web应用可以在离线状态下依然可访问，在离线状态下使用LocalStorage保存相关数据，同时还有一套同步机制，在再次回到在线状态时，能将存储的数据传输到服务器，同时客户端同步最新的存储内容。

检查网络是否在线：navigator.onLine

原理：

1. 浏览器访问离线应用，查看页面中是否有清单文件，如<html manifest=”example .appcache”>。有的话就返回cache中的资源。
2. 在线状态下，浏览器会异步检测服务器上的manifest文件，如果manifest文件有更新，新的清单文件以及清单中列出的所有文件都会下载保存到浏览器的应用程序缓存中。
3. 如果没有更新，则不需要拉取资源文件。  
   注意：其实他有两个缺点。一是它只是检查manifest文件，如果某个资源文件有更新，如果想让这个文件生效，那么你必须去修改manifest文件。除此之外，不需要第二次刷新才能让最新的版本生效。

1. 创建manifest文件。  
   CACHE MANIFEST  
   #version  
     
   CACHE:  
   #表示要缓存的资源文件，相对路径的话也是相对清单manifest文件  
   /CSS/sample.css  
     
   NETWORK:  
   #B表示必须走网络拉取的资源，如果写\*表示除了CACHE中的全部都要走网络  
   \*  
     
   FALLBACK：  
   #每行都包括两个URL。第二个URL指需要加载和存储在缓存中的资源，第一个URL是前缀，凡是匹配到第一个URL前缀的资源，当走网络获取失败的时候，都是用第二个URL指定的资源代替  
   / offline.html
2. Manifest文件后缀名为.appcache,可以是绝对路径，也可以是相对路径，但当是决定路径的时候，必须同源，由此可看客户端存储都有一个同源限制
3. 服务器上添加 MIME类型为 text/cache-manifest.
4. 如果想让缓存“卸载”，则从服务器端删除manifest文件，同时修改HTML代码以便他们与manifest文件断开连接。  
   异步：

客户端缓存都是注册在window对象的ApplicationCache对象上。  
本页一定会缓存。

1.每次加载带有manifest的HTML文件，都会触发checking事件。

2.如果manifest文件没有更新，则触发noupdate事件。

3.如果manifest文件有更新，则依次触发downloading事件、progress事件、

undateready事件。

4.如果首次加载，一样会触发checking事件，随后downloading、progress、cached。

5.如果离线，就无法检查清单文件，触发error事件。  
6. 如果检查的清单文件在服务器上已不存在，是404，则也会触发obsolute事件，并将缓存“卸载”删除。  
  
  
“卸载”的时候不仅要将<html>中的manifest属性去掉，还要删除客户端的manifest文件，缺一不可。

因为manifest文件没有更新，客户端依然会加检测服务器端的manifest文件。  
依然会触发checking、obsolute事件。因为他会检测sever端有没有manifest文件，如果没有，那就他谈不上更新不更新。走网络，强制让manifest失效。  
  
  
  
  
如何更新？

如果有修改资源文件，必须要修改manifest文件。有可能会觉得不太方便。

优点？

1.完全离线  
2.资源被缓存，加载更快

3.降低服务器端负载。  
缺陷？  
1.含有manifest属性的当前请求页无论如何都会被缓存。  
2.需要两次刷新才能获取新资源，因为它的更新是完全异步的。

3.更新是全局性的，无法单独更新某个文件，全部拉取。  
4.对于连接的参数是敏感的，任何一个参数的修改都会被（master）重新缓存。如index.html和index.html?uid=yuqi会被认为是不同文件，会被分别缓存。  
  
所以说，链接上有参数的话不建议使用appcache。  
  
应用场景？

1.单地址的页面

2.实时性要求不高的业务

3.离线app